A New Fast Large Neighbourhood Search for Service Network Design with Asset Balance Constraints

Ruibin Bai
School of Computer Science
University of Nottingham Ningbo China
Telephone: +86-574-88180278
Email: ruibin.bai@nottingham.edu.cn

John R. Woodward
School of Computing
Science and Mathematics
University of Stirling, Scotland.
Email: john.woodward@cs.stir.ac.uk

Nachiappan Subramanian
Nottingham University Business School
The University of Nottingham Ningbo China
Email: nachiappan.subramanian@
nottingham.edu.cn

***Abstract*—The service network design problem (SNDP) is a fundamental problem in consolidated freight transportation. It involves the determination of an efficient transportation network and the scheduling details of the corresponding services. Compared to vehicle routing problems, SNDP can model transfers and consolidations on a multi-modal freight network. The problem is often formulated as a mixed integer programming problem and is NP-Hard. In this research, we propose a new efficient large neighbourhood search function that can handle the constraints more efficiently. The effectiveness of this new neighbourhood is evaluated in a tabu search metaheuristic (TS) and a GLS guided local search (GLS) method. Experimental results based on a set of well-known benchmark instances show that the new neighbourhood performs significantly better than the previous arc-flipping neighbourhood. The neighbourhood function is also applicable in other optimisation problems with similar discrete constraints.**

I. Introduction

The SNDP is widely considered as the core problem of the freight transportation planning for less-than truck load transport and express deliveries where consolidation is necessary to improve the efficiency. It involves the determination of a cost-effective transportation network and the services which it will provide, while satisfying the constraints related to demand diversities both geographically and temporally, network availability, assets capacity, etc. The SNDP is strongly NP-Hard (Ghamlouche et al., 2003) and hence solving the problem of decent sizes to optimality is generally not practical. In fact, the SNDP is generally of large-scale, due to the size of potential network (defined by the number of nodes, the number of arcs and the number of commodities to be delivered). This is particularly the case when the formulation is based on a time-space network in which each node and each arc has a copy in each period of the scheduling horizon (see Fig. 1).

Various heuristic and metaheuristic approaches have been applied to this problem and substantial progress has been made (Crainic and Kim, 2007; Bai et al., 2012). However, most current research in this area has focused on various intelligent high-level search strategies but analysis of the problem solution structure and its constraints is very limited. These approaches do not satisfy real-world requirements either in solution quality or in computational time. This is because the SNDP contains some difficult constraints and a flow distribution subproblem, generally referred as the capacitated multicommodity min-cost flow (CMMCF) problem, which is very expensive to solve with an iterative metaheuristic approach. This motivates us to develop more efficient metaheuristics for this important and challenging problem. In this paper, unlike the other approaches, we proposed and studied a new larger neighbourhood that exploits the special structure of the SNDP constraints and has much better reachability due to the implicit constraint handling. The experiments on two basic metaheuristic approaches show that the new neighbourhood is very effective and could be used to develop more efficient algorithms for the SNDP.

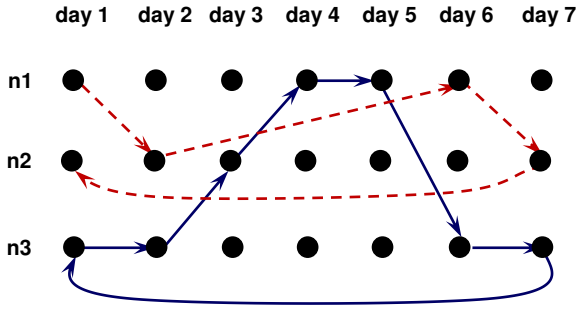


Fig. 1: An example of a time-space network with 3 nodes and 7 periods.

The remainder of the paper is structured as follows: section II provides a brief introduction to the SNDP and an overview of the research in freight service network design. Section III presents the arc-node based mathematical formulation for SNDP and section IV discusses the neighbourhood structure used in the previous studies. Section V describes the proposed k-node neighbourhood operator whose performance is evaluated in Section VI through a TS method and a GLS method. Section VII concludes the paper.

II. Problem Description and Literature Review

The SNDP is an important tactical/operational freight transportation planning problem. It is of particular interest for less-than truck load transport transportation and express delivery services, where consolidation of deliveries is widely adopted in order to maximise the utilisation of freight resources (Crainic, 2000). The SNDP involves the search for optimal or near-optimal service characteristics, including the selection of routes and the vehicle types for each route, the service frequency and the delivery timetables, the flow distribution paths for each commodity, the consolidation policies, and the idle vehicle re-positioning, so that legal, social and technical requirements are met (Wieberneit, 2008).

The SNDP differs from the capacitated multicommodity network design (CMND) problem, a well-known NP-Hard problem, in that it has an additional source of complexity due to the required *balance constraint* for freight assets in order to ensure that vehicle routes are contiguous and that vehicles are in the correct positions after each planning cycle. The presence of both this constraint and the flow conservation constraint makes the problem extremely challenging for many neighbourhood based optimisation methods. The remainder of this section provides a brief overview of the previous research into SNDP. More comprehensive reviews can be found in (Crainic, 2000; Crainic and Kim, 2007; Wieberneit, 2008).

Service network design is closely related to the classic network flow problems (Ahuja et al., 1993). Early work in this field includes (Crainic and Rousseau, 1986; Powell, 1986). Crainic et al. (2000) investigated a hybrid approach for CMND, combining a TS method with pivot-like neighbourhood moves and column generation. Ghamlouche et al. (2003) continued the work and proposed a more efficient cycle-based neighbourhood structure for CMND. Experimental tests, within a simple TS framework, demonstrated the superiority of the method to the earlier pivot-like neighbourhood moves in (Crainic et al., 2000). This approach was later enhanced by adopting a path-relinking mechanism (Ghamlouche et al., 2004).

Barnhart and her research team (Barnhart et al., 2002) addressed a real-life air cargo express delivery SNDP. The problem instances are characterised by their large sizes and the addition of further complex constraints to those which are in existence in the general SNDP model. A `tree formulation` was introduced and the problem was solved heuristically using a method based upon column generation. Armacost et al. (2002) introduced a new mathematical model based on an innovative concept called the `composite variable`, which has a better Linear Programming (LP) bound than other models. A column generation method using this new model was able to solve the problem successfully within a reasonable computational time, taking advantage of the specific problem details. However, it may be difficult to generalise the model to other freight transportation applications, especially when there are several classes of services being planned simultaneously.

Pedersen et al. (2009) studied more generic SNDP in which the asset balance constraint was present. A multi-start metaheuristic, based on TS, was developed and tested on a set of benchmark instances. The TS method outperformed a commercial MIP solver when computational time was limited to one hour per instance on a test PC with a Pentium IV 2.26GHz CPU. Andersen et al. (2009) compared the node-arc based formulation, the path-based formulation and a cycle-based formulation for SNDPs. Computational results on a set of small randomly generated instances indicated that the cycle-based formulation gave significantly stronger bounds than the other two and hence may allow for much faster solution of problems. More recent work by Bai et al. (2012) attempted to further reduce the computational time and investigated a guided local search based (GLS) hybrid approach. The computational study, based on a set of popular benchmark instances, showed that GLS, if configured appropriately, was able to obtain better solutions than TS but with less than two thirds of the computational time. However, GLS in that study was based on an arc-flipping neighbourhood which sometimes leads to poor solutions. Barcos et al. (2010) investigated an ant colony optimisation approach to address a (simplified) variant of freight SNDP. The algorithm was able to obtain solutions better than those adopted in the real-world within a reasonable computational time. Andersen et al. (2011) studied a branch and price method for the SNDP. Although the proposed algorithm was able to find solutions of higher quality than the previous methods, the 10-hour computational time required by the algorithm poses a great challenge for practical applications.

It can be seen that very limited research has been done to investigate more on efficient neighbourhood structures to tackle the difficult constraints and expensive flow distribution subproblems. The goal of this paper is to address this gap by studying a new neighbourhood structure for SNDP. The effectiveness of the new structure is evaluated in two metaheuristic approaches (TS and GLS) for a set of well-known SNDP benchmark instances.

III. The Freight SNDP Model

The problem of concern in this paper can be formulated in several ways. We used a node-arc based model described in (Pedersen et al., 2009) and also present it here for completeness. The list of notation used in the model is given in Table I.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ denote a directed graph with nodes $\mathcal{N}$ and arcs $\mathcal{A}$. Let (i, j) denote the arc from node i to node j. Let $\mathcal{K}$ be the set of commodities. For each commodity $k \in \mathcal{K}$, let $o(k)$ and $s(k)$ denote its origin and destination nodes, respectively. Let y_{ij} be boolean decision variables, where y_{ij} = 1 if arc (i, j) is used in the final design and 0 if it is not used. Let x_{ij}^k denote the flow of commodity k on arc (i, j). Let u_{ij} and f_{ij} be the capacity and fixed cost, respectively, for arc (i, j). Finally, let c_{ij}^k denote the variable cost of moving one unit of commodity k along arc (i, j). The SNDP can then be formulated as follows:

TABLE I: List of notations used in the SNDP model

Notation	Meaning
$\mathcal{N}$	The set of nodes.
$\mathcal{A}$	The set of arcs in the network.
$\mathcal{G} = (\mathcal{N}, \mathcal{A})$	Directed graph with nodes $\mathcal{N}$ and arcs $\mathcal{A}$.
$(i,j) \in \mathcal{A}$	The arc from node i to j.
u_{ij}	Capacity of arc (i,j).
f_{ij}	The fixed cost of arc (i,j).
$\mathcal{K}$	The set of commodities.
$o(k)$	The origin (source) of commodity $k \in \mathcal{K}$.
$s(k)$	The sink (destination) of commodity k.
d^k	The flow demand of commodity k.
c_{ij}^k	The variable cost for shipping a unit of commodity k on the arc (i,j).
x_{ij}^k	The amount of flow of commodity k on the arc (i,j).
y_{ij}	The network design variables. $y_{ij} = 1$ if arc (i,j) is open and 0 if it is closed.
$\mathbf{x}$	The vector of all flow decision variables, i.e. $\mathbf{x} =< x_{00}^0, \ldots, x_{ij}^k, \ldots >$.
$\mathbf{y}$	The vector of all design variables, i.e. $\mathbf{y} =< y_{00}, ..., y_{ij}, ... >$.
$\mathcal{N}^+(i)$	The set of outward neighbouring nodes of node i.
$\mathcal{N}^-(i)$	The set of incoming neighbouring nodes of i.
b_i^k	The outward flow of commodity k. $b_i^k = d^k$ if $i = o(k)$, $b_i^k = -d^k$ if $i = s(k)$ and 0 otherwise.
$z(\mathbf{x}, \mathbf{y}), z(s)$	The objective of SNDP model, which represents the sum of the fixed cost and the variable cost for given solution vectors $\mathbf{x}$ and $\mathbf{y}$, or expressed in terms of a potential solution s.
$g(s)$, $g(\mathbf{x}, \mathbf{y})$	The objective function which is actually solved, including a penalty for infeasibility, expressed in terms of a potential solution s or the decision variable component vectors $\mathbf{x}$ and $\mathbf{y}$ of s.

$$\min \quad z(\mathbf{x}, \mathbf{y}) = \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \qquad (1)$$

subject to

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij} y_{ij} \quad \forall (i,j) \in \mathcal{A} \qquad (2)$$

$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^k - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^k = b_i^k, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K} \qquad (3)$$

$$\sum_{j\in\mathcal{N}^-(i)} y_{ji} - \sum_{j\in\mathcal{N}^+(i)} y_{ij} = 0 \quad \forall i \in \mathcal{N} \qquad (4)$$

where $x_{ij}^k \geq 0$ and $y_{ij} \in \{0, 1\}$ are the decision variables. The network capacity constraint (2) ensures that the maximum capacity of arc (i,j) is not violated. The flow conservation constraint (3) ensures that the entire flow of each commodity is delivered to its destination, where $\mathcal{N}^+(i)$ denotes the set of outward neighbours of node i and $\mathcal{N}^-(i)$ the set of inward neighbours. b_i^k is the outward flow of commodity k for node i, so we set $b_i^k = d^k$ if $i = o(k)$, $b_i^k = -d^k$ if $i = s(k)$, and $b_i^k = 0$ otherwise. Constraint (4) is the *asset-balance constraint*, which is missing from the standard CMND formulation, as discussed in section II and which ensures the balance of transportation assets (i.e. vehicles) at the end of each planning period.

For a given design variable vector $\overline{\mathbf{y}} =< \overline{y_{00}}, ..., \overline{y_{ij}}, ... >$, the problem becomes one of finding the optimal flow distribution variables. Constraint (4) is no longer relevant and the flow must be zero on all closed arcs, so only open arcs have to be considered in the model. Let $\overline{\mathcal{A}}$ denote the set of open arcs in the design vector $\overline{\mathbf{y}}$, then flow distribution variables (x_{ij}^k) for all open arcs $((i,j) \in \overline{\mathcal{A}})$ can be obtained by solving the following capacitated CMMCF problem, where $x_{ij}^k \geq 0$ $\forall (i,j) \in \overline{\mathcal{A}}$, k:

$$\min \quad \overline{z}(\mathbf{x}) = \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\overline{\mathcal{A}}} c_{ij}^k x_{ij}^k \qquad (5)$$

subject to

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij} \quad \forall (i,j) \in \overline{\mathcal{A}} \qquad (6)$$

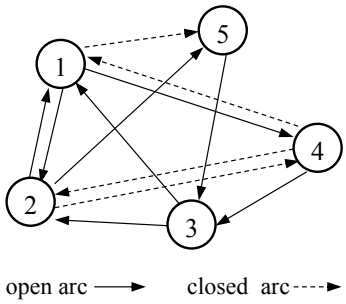
$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^k - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^k = b_i^k, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K} \qquad (7)$$

IV. A Revisit of Previous Heuristic Approaches

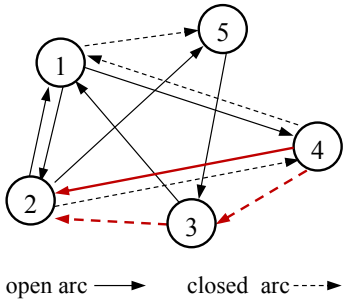
In the previous efforts (Pedersen et al., 2009; Bai et al., 2012), neighbourhood search functions were primarily based on single arc state-flipping (or referred as arc adding/dropping) with the flow distribution handled separately either heuristically (based on a residual graph) or optimally by solving the corresponding capacitated CMMCF problem using an LP solver. Interested reader is referred to (Pedersen et al., 2009) for more details of this neighbourhood structure.

However, one drawback of this neighbourhood is its inability of maintaining solution feasibility in terms of asset-balance constraints. For a feasible solution satisfying the asset-balance constraints, flipping the state of a single arc will typically generate an infeasible solution (i.e. violating constraint (4)). Let us take a simple network in Fig. 2 as an example. In the current configuration (Fig. 2.(a)), the network consists of 8 open arcs (and 4 closed arcs) and is asset-balanced since, for each node, the number of incoming arcs equals to the number of outgoing arcs. Using the neighbourhood function in (Pedersen et al., 2009; Bai et al., 2012), one could generate 12 neighbouring solutions. Unfortunately none of them is feasible due to asset balance constraint violations. For example, opening arc (1,5) will lead to vehicle imbalance at both nodes 1 and 5. Similarly, closing arc (2,1) will lead to asset-balance constraint violations at nodes 1 and 2. In (Pedersen et al., 2009; Bai et al., 2012), this constraint violation issue was addressed by using a special feasibility-recovery procedure at the end of each local search phase. Although effective in finding a feasible solution, the method may suffer from performance issues when feasibility-recovery procedure leads to large increase in costs, and hence inferior solutions.

Another major drawback of the arc-flipping neighbourhood function is the reachability in the search space. Observations from experimental tests in (Bai et al., 2012) show that considerable number of neighbourhood moves are rejected during the search and local search methods (both TS and



(a) Current solution



(b) A neighbouring solution

Fig. 2: An illustration of the reachability issue of the arc-flipping neighbourhood. The statuses of the thicker arcs are changed during the neighbourhood move.

GLS) tend to get stuck at local optima. It appears that this neighbourhood function struggles to reach certain regions of the search space regardless of the number of iterations permitted. This observation explains why the "multiple starts" used in (Pedersen et al., 2009; Bai et al., 2012) is effective. In fact, this can be illustrated by the network in Fig. 2. Assume that the network shown in Fig. 2.(b) is a better feasible solution than Fig. 2.(a). Moving from the solution in Fig. 2.(a) to the solution depicted in Fig. 2.(b) requires closing two arcs 4→3 and 3→2 and opening arc 4→2. Since only once arc can be modified at each neighbourhood move (excluding arcs that are modified during the flow redistribution procedure), in theory it is possible to move to the neighbouring solution in Fig. 2.(b) through 3 successive operators. In practice the success rate of such a move could be extremely low since the first two moves will result in asset imbalance at all three nodes involved and the penalty for this constraint violations can prevent the intermediate solutions from being accepted. In addition, if the flow redistribution during any of these three moves is infeasible, the search will not reach solution depicted in Fig. 2.(b) from Fig. 2.(a). This explains why the multi-start was required in the previously proposed algorithms.

V. The Proposed k-node Neighbourhood

In this section, we describe the proposed new neighbourhood which was originated from the idea of paired-route-flipping. The main purpose is to maintain the feasibility of

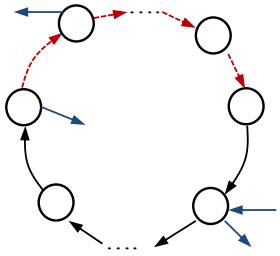


Fig. 3: An illustration of the paired route-flipping neighbourhood. Solid lines are open arcs and dashed ones are closed arcs.

the solution during the search by changing the statuses of two carefully selected routes. Each route is a sequence of arcs representing vehicle moves over time. We describe this idea in the following subsection.

A. The paired route-flipping

In particular, instead of flipping an arc, we identify a set of arc-flipping operations with automatic feasibility satisfaction in terms of the asset-balance constraint. Fig. 3 illustrates this arc-flipping operator. The solid lines represent open arcs and dotted arcs denote closed arcs. The paired-route-flipping operator involves simultaneously changing the statuses of two routes which form a cycle but have opposite statuses (i.e. one closed and one open). Although this neighbourhood operator can guarantee satisfaction of asset-balance constraints, identifying such a pair of routes is not trivial. On contrary, it is much easier to focus on nodes rather than arcs, leading to our k-node neighbourhood structure which we describe in the next subsection.

B. The k-node neighbourhood

In this neighbourhood, a subset of k nodes out of all the nodes are selected and arcs incident upon these nodes are considered for changes. Note that in order to prevent evaluating a candidate solution many times, we require that the change of arcs should involve exactly k nodes rather than a subset of them. We focus on the small and medium sized neighbourhoods. Large neighbourhoods (e.g. $k > 4$) are not considered since it is impractical to evaluate them.

Fig. 4 illustrates the k-node operator when k = 2, 3 and 4 respectively. It is not difficult to see that when k=2, a feasible neighbour may exist only if both arcs connecting the two nodes have a same status (i.e. either both closed or both open). If one of them is open and the other is closed, no feasible neighbouring solution exists.

When $k = 3$, the maximum number of arcs between these nodes is 6. For a feasible current solution s, we denote design variables for arcs from $a_0, a_1, ..., a_5$ as $y_0, y_1, ..., y_5$ respectively. Including the current solution s, the maximum number of neighbouring solutions for 3-node operator will be $2^6 = 64$. However, not every neighbouring solution will be feasible in terms of asset-balance constraint (4). For any

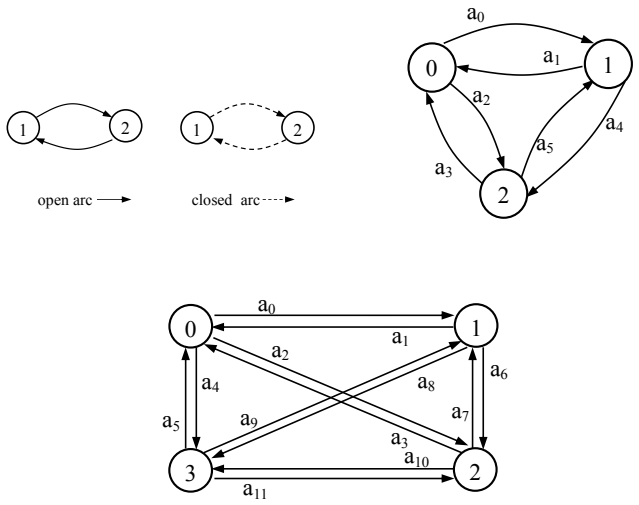


Fig. 4: An illustration of k-node operator when $k = 2$, 3, and 4 respectively.

neighbouring solution s', to satisfy asset-balance constraint, the following constraints should be respected, each of which corresponds to one of the three nodes under consideration (we denote the corresponding design variables in s' for arcs $a_0, a_1, ..., a_5$ as $y'_0, y'_1, ..., y'_5$ respectively).

$$y_0 + y_2 - y_1 - y_3 = y'_0 + y'_2 - y'_1 - y'_3 \quad (8)$$

$$y_1 + y_4 - y_0 - y_5 = y'_1 + y'_4 - y'_0 - y'_5 \quad (9)$$

$$y_3 + y_5 - y_2 - y_4 = y'_3 + y'_5 - y'_2 - y'_4 \quad (10)$$

Note that any of the two conditions will be sufficient to ensure feasibility since the third condition can be obtained from the other two conditions. Taking into account these two conditions, the maximum number of asset-balanced neighbouring solutions for s will be 61 = (63-2).

Similarly one can work out the 4 conditions to be satisfied for candidate solutions when $k = 4$. Again, only 3 out the above 4 conditions are active and the other one is redundant. For a medium sized network of 60 nodes, the number of subsets of nodes with cardinality of 4 is $C^4_{60} = 487635$. For each node subset, the maximum size of the neighbourhood is $2^{12} - 3 = 4096$ since there are maximum of 12 arcs between these nodes and 3 active constraints. Considering the time taken to solve the flow distribution subproblem for each of these candidate solutions in the neighbourhood, it is impractical to efficiently evaluate neighbourhoods larger than $k = 4$. Even with $k = 4$, it could still be very slow to have complete evaluation of the candidate solutions. Faster neighbourhood search procedures are required.

C. Speeding Up the Neighbourhood Search

In this section, we discuss ways that could speed up the neighbourhood search. In the previous neighbourhood structure, there may be solutions which can be discarded directly without ascertaining their objective values. Firstly, given a solution s and one of its neighbouring solutions s', if too many arcs are closed in s' compared to s, there is very little chance that the flow on these arcs can be redistributed among the remaining network. It is therefore not necessary to solve the CMMCF sub-problem. Similarly if a neighbouring solution s' has too many open arcs than the original solution, it is not necessary to evaluate this solution either since the fixed cost would increase dramatically, resulting in a poor solution. These two "extreme" cases are dealt by adding cut-set inequalities and a heuristic rule respectively which we now discuss

Let N_k be the set of k nodes selected in the k-nodes neighbourhood and A_k be the set of arcs that are participated by any of two nodes from $N-K$. For a given k, the maximum number of arcs incident with these k nodes is $P^2_k = k(k-1)$. For each of node $i \in N_k$, we define the following cut-sets S_i and T_i:

$$S_i = \{N_k \backslash i\}, T_i = \{N \backslash S_i\} \quad (11)$$

Then the following aggregate capacity demand inequalities should be satisfied.

$$Cap_{st} > Demand_{st} \quad \forall s \in S_i, \forall t \in T_i, \forall i \in N_k \quad (12)$$

$$Cap_{ts} > Demand_{ts} \quad \forall s \in S_i, \forall t \in T_i, \forall i \in N_k \quad (13)$$

In addition, the following set-cuts are defined the corresponding inequalities are valid for each node $i \in N_k$.

$$S'_i = N^+_i, \quad T'_i = \{N \backslash S'_i\} \quad (14)$$

$$Cap_{s't'} > Demand_{s't'} \quad \forall s \in S'_i, \forall t \in T'_i, \forall i \in N_k \quad (15)$$

$$Cap_{t's'} > Demand_{t's'} \quad \forall s \in S'_i, \forall t \in T'_i, \forall i \in N_k \quad (16)$$

In the case of an "excessive" number of open arcs in s' compared to s, the following condition is used to check whether s' will be evaluated or discarded. Neighbours that do not satisfy this condition will be discarded.

$$\sum_{a \in A_k} f_a \times y'_a \leq \sum_{a \in A_k} f_a \times y_a + 2 \times \overline{f_k} \quad (17)$$

where $\overline{f_k}$ is the average fixed cost of the arcs in A_k that are involved in this neighbourhood move. (too informal) We discard a neighbouring solution if it contains 2 more open arcs than the original solution, evaluated in terms of the average fixed costs.

The number of nodes required for k-node neighbourhood is at least 2. For a given input k (≥ 2), a neighbouring solution can be generated by making changes to arcs participated by exactly h ($2 \leq i \leq k$) nodes.

For every neighbouring design variable vector $\mathbf{y}'$, the procedure first checks whether asset-balance constraint is respected by this vector. If not, $\mathbf{y}'$ is discarded and the next vector is considered. The asset-balance constraint is checked in the following way. When $h = 2$, as discussed in the previous section, $\mathbf{y}'$ is feasible only if two arcs participated by the two nodes have a same status (i.e. both open or close). When $h = 3$ or $h = 4$, one can check the asset-balance at each node using conditions (8-10) and (11-14) respectively. When $k > 4$, the size of the neighbourhood increases significantly. It is impractical to evaluate the neighbourhood when $k > 4$ because of expensive CMMCF solution calls required in the neighbourhood search. Therefore in our experiment, we set $k = 4$.

Once a neighbouring design variable vector $\mathbf{y}'$ satisfies the asset-balance constraint, we check it against the inequality conditions (16-21) to filter design variable vectors that are deemed either infeasible or less interesting. After this, the CMMCF procedure is called to find a feasible flow if it exists. If the corresponding node set ns is tabued and aspiration criterion is not met, this solution is overlooked. Otherwise, it is compared against the initial solution and best solution so far. If a candidate solution improves the initial solution, the procedure returns the first-improvement solution. Otherwise, it returns the best solution (s^*) in the current neighbourhood.

VI. Performance Evaluation

In this section, we evaluate the performance of the k-node neighbourhood against the current best approaches based on the arc-flipping neighbourhood. We chose tabu search and guided local search for comparisons as both approaches have previously been applied to this problem and have shown good performance.

A. A Tabu search approach with k-node neighbourhood

We firstly implement a basic TS method with the proposed k-node neighbourhood function (denoted as TS_k-node) to evaluate its performance. We compare it against a multi-start TS method given in (Pedersen et al., 2009) and a tabu assisted multi-start GLS method described in (Bai et al., 2012). Both algorithms use the arc-flipping neighbourhood function.

The pseudo-code of TS_k-node algorithm is given in Algorithm 1. The inputs of the algorithm include a feasible initial solution s_0, the objective function of the problem $z(.)$, the maximum number of nodes allowed in the neighbourhood generation k, and the maximum length of the tabu list TL. In our experiment, we set $k = 4$ to keep the size of the neighbourhood relative small so that it can be evaluated relatively efficiently. We used a fixed length tabu list `TabuList` which is maintained on the first-in-last-out basis. The maximum length is set to $TL = 20$ after some initial tests on a subset of the benchmark instances. Because the proposed *k-node* neighbourhood is based on node sets rather than arcs, the tabu list contains the node set which leads to the adoption of the current solution returned by the procedure `FirstDescent(`$s', z(.), k$`)`. The procedure

Algorithm 1 A basic TS with k-node neighbourhood

input An initial feasible solution s_0, the objective function $z(.)$, k, tabu length TL.
Initialise the TabuList, the current solution $s' = s_0$, and the best solution $s_b = s_0$.
while stopping criterion is not met **do**
  $s', ns \leftarrow$ FIRSTDESCENT$(s', z(.), k)$ ▷ Get the first-descent solution and the node set ns
  if $z(s') < z(s_b)$ **then**
    $s_b = s'$ ▷ Update the best solution
  end if
  TabuList.Add(ns) ▷ Add the corresponding node set to the TabuList
  if (TabuList.Length $> TL$) **then**
    TabuList.RemoveFrist ▷ Maintain the TabuList
  end if
end while
return s_b

repeatedly calls the `FirstDescent(.)` to search for a first-decent neighbouring solution which is not in the tabu list until the stopping criterion is met. In this case, the procedure stops when the maximum allowed time is exhausted. This was set to 2400s minus the amount of time taken to create the initial feasible solution.

B. A guided local search with k-node neighbourhood

We also implemented a basic GLS method with the proposed neighbourhood. The pseudo-code of the algorithm is given in Algorithm 2. GLS is a metaheuristic designed for constraint satisfaction and combinatorial optimisation problems (Voudouris and Tsang, 2003). The underlining idea is to take advantage of information gathered during the search to guide it and enable it to escape local optima. GLS adopts a transform objective function which includes a penalty to penalise "unattractive" features in a candidate solution. We denote p_r as the current penalty for the presence of a given feature r in the current solution s, and $I_r(s)$ is an indicator variable such that $I_r(s) = 1$ if the candidate solution s contains feature r and $I_r(s) = 0$ otherwise. λ is a scaling parameter between the original objective function $z(s)$ and the aggregated feature penalties. Since λ is problem instance dependent and is difficult to tune directly, it was estimated by $\lambda = \alpha g(s^*)/\sum_r I_r(s^*)$ where s^* is the current best solution and α is a parameter that is less problem-dependent than λ. At each GLS iteration, the proposed k-node neighbourhood function `FirstDescent(.)` was used to find a first-descent solution except that the TabuList in `FirstDescent(.)` was set to empty.

In order to have a fair comparison, both the TS and GLS start from a same initial feasible solution for each problem instance. Similar to TS_k-node, the stopping criterion was 2400 seconds of computational time, minus the time spent for generating an initial feasible solution and the size of the neighbourhood is set to $k = 4$. The GLS parameter was set to

Algorithm 2 Pseudo-code for a basic guided local search with new neighbourhood function.

input an initial feasible solution s_0, an original objective function $z(s)$, a set of features R, the cost h_r associated with each feature $r \in R$ and a scaling parameter λ.
output an improved solution s'.

```
foreach r ∈ R, set p_r := 0
initialise s ← s_0 and I_r(s), set g(s) = z(s) + λ × Σ_r p_r I_r(s)
while stopping criterion is not met do
    s ← FIRSTDESCENT(s, g(s), k)        ▷ Get the first descent solution with regard to g(s)
    for all r ∈ R do
        util_r(s) = I_r(s) × h_r / (1 + p_r)
        Find r with maximum util_r, set p_r ++
    end for
end while
return s' ← best solution found according to the original objective function z(s).
```

$\alpha = 0.05$ based on a preliminary experiment on a number of representative instances.

C. Computational results and discussions

TABLE II: Computational results for a TS_k-node and GLS_k-node, in comparison with two previous algorithms; TS_arc-flip (Pedersen et al., 2009) and TS_MGLS (Bai et al., 2012).

Inst.	TS_arc-flip	TA_MGLS	TS_k-node	GLS_k-node
c37	102919	99622	**98498**	98567
c38	150764	143867	**142770**	143190
c39	103371	102833	**101931**	103010
c40	149942	143839	**141475**	147209
c45	82533	**79895**	80032	80355
c46	128757	**124454**	124873	126474
c47	78571	**78302**	78393	79282
c48	116338	**115836**	115939	118838
c49	55981	55986	**55551**	56137
c50	104533	**102017**	102838	103662
c51	54493	54708	**54177**	54642
c52	105167	105423	**105047**	106833
c53	119735	**116915**	117638	117570
c54	162360	**156008**	157810	157925
c55	120421	**118894**	119609	119470
c56	161978	**159427**	160096	160162
c57	49429	49457	**49210**	49271
c58	63889	**62774**	62947	63503
c59	48202	47728	**47477**	47738
c60	58204	58046	**58015**	58408
c61	103932	**102216**	102391	102356
c62	157043	**144755**	145397	145148
c63	103085	**99726**	100099	100019
c64	141917	**136727**	137518	136795

Table II presents the computational results by the TS and GLS with the proposed neighbourhood function in comparison with the latest metaheuristics for this problem. TS in (Pedersen et al., 2009) was run on a Pentium IV 2.26GHz PC with 3600 seconds CPU time. All others were run on a PC with 1.8GHz Intel Core 2 CPU, single-threaded and a 2400-second time limit in conjunction with CPlex12 as the linear programming solver. The experiments were based on a set of benchmark instances drawn from (Pedersen et al., 2009). This data set consists of 24 instances of different sizes (nodes, arcs, commodities) and distributions of fixed cost, variable cost and capacity. From instance c37 to c64, the problem size increases steadily. For each instance, 10 independent runs were conducted and their average results are reported. The best results among the three approaches are highlighted in bold. It can be seen that even within very basic TS and GLS framework, the new neighbourhood function is able to produce very competitive results. Both the TS method in (Pedersen et al., 2009) and the tabu assisted multi-start GLS method (TA_MGLS) in (Bai et al., 2012) used a multi-start framework to diversify the search. It can be seen that the proposed neighbourhood evaluated in a basic TS, performed better than the TS method in (Pedersen et al., 2009). It also outperformed TA_MGLS for many instances, particularly small instances (c37-c40). For large instances (e.g. c61-c64), TS_k-node was slightly inferior to TA_MGLS. This is probably caused by longer computational time taken by each `FirstDescent(.)` procedure call for larger sized problems which leads to significant increase in CMMCF solution time. A possible improvement for this algorithm is then to develop some heuristic flow distribution procedure to reduce the number of CMMCF calls. In terms of the performance of GLS_k-node, it outperformed the multi-start TS with arc-flipping neighbourhood for 18 out 24 instances. Compared with TA_MGLS which is much sophisticated, GLS_k-node was inferior for most instances but obtained better results for instance C51 and C57. GLS_k-node is generally competitive when the problem size is small. For large instances, each `FirstDescent(.)` call is expensive and hence impedes the search significantly. This is compounded with influence of the transformed objective function $g(s)$ used in GLS that leads to poor solutions since local optima were not reached when the computational time is not sufficient. This also explains why TS_k-node was able to obtain better solutions than GLS_k-node in general although both of them started from the same initial solutions and used exactly the same neighbourhood function. Nevertheless, through these two experiments, the new neighbourhood function has shown it effectiveness by producing very promising results, obtaining the new best-known results for many instances. This is largely attributed to its better reachability because of larger neighbourhood sizes and abilities to maintain feasibility. Contrary to many other neighbourhood operators, the proposed new neighbourhood operator uses the constraint violations to their advantage by ignoring lots of infeasible solutions. Compared with the previous neighbourhood function, the superiority of the k-node operator was demonstrated by the superior results obtained both by the basic GLS and basic TS without the multi-start mechanism which was crucial in a previous hybrid method TA_MGLS in order to prevent the search from getting stuck at local optima.

VII. Conclustions and Future Work

Service network design is the core problem for freight transportation network planning and optimisation. The problem is strongly NP-Hard and is particularly challenging due to the problem scale and complex constraints. Differing from the previous studies for this problem which have been focusing on more effective top-level search strategies, this research proposed and studied a novel neighbourhood structure that permits simultaneous changes of multiple arcs incident upon k given nodes but maintains the solution feasibility throughout the search. The new neighbourhood, evaluated in the context of two metaheuristic approaches, showed better reachability than the existing arc-flipping neighbourhood and produced some of the best known results for a set of benchmark instances. As the proposed neighbourhood can also be taken advantage of in conjunction with more sophisticated metaheuristics and even better results are therefore anticipated. In addition, the proposed neighbourhood structure could be used for other network problems with similar discrete equality constraints.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant No. 71471092), Zhejiang Natural Science Foundation (Grant No. LR17G010001) and Ningbo Science and Technology Bureau (Grant No. 2014A35006).

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows: Theory, Algorithms and Applications. Prentice Hall.

Andersen, J., Christiansen, M., Crainic, T. G., Gronhaug, R., 2011. Branch and price for service network design with asset management constraints. Transportation Science 45 (1), 33–49.

Andersen, J., Crainic, T. G., Christiansen, M., 2009. Service network design with management and coordination of multiple fleets. European Journal of Operational Research 193 (2), 377 – 389.

Armacost, A. P., Barnhart, C., Ware, K. A., 2002. Composite variable formulations for express shipment service network design. Transportation Science 36 (1), 1–20.

Bai, R., Kendall, G., Qu, R., Atkin, J., 2012. Tabu assisted guided local search approaches for freight service network design. Information Sciences 189, 266–281.

Barcos, L., Rodriguez, V., Alvarez, M., Robuste, F., 2010. Routing design for less-than-truckload motor carriers using ant colony optimization. Transportation Research Part E 46, 367–383.

Barnhart, C., Krishnan, N., Kim, D., Ware, K., 2002. Network design for express shipment delivery. Computational Optimization and Application 21, 239–262.

Crainic, T. G., 2000. Service network design in freight transportation. European Journal of Operational Research 122 (2), 272 – 288.

Crainic, T. G., Gendreau, M., Farvolden, J. M., 2000. A simplex-based tabu search method for capacitated network design. INFORMS Journal on Computing 12 (3), 223–236.

Crainic, T. G., Kim, K. H., 2007. Chapter 8 intermodal transportation. In: Barnhart, C., Laporte, G. (Eds.), Transportation. Vol. 14 of Handbooks in Operations Research and Management Science. Elsevier, pp. 467 – 537.

Crainic, T. G., Rousseau, J.-M., 1986. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. Transportation Research Part B: Methodological 20 (3), 225 – 242.

Ghamlouche, I., Crainic, T. G., Gendreau, M., 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. Operations Research 51 (4), 655–667.

Ghamlouche, I., Crainic, T. G., Gendreau, M., 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. Annals of Operations Research 131, 109–133.

Pedersen, M. B., Crainic, T. G., Madsen, O. B., 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. Transportation Science 43 (2), 158–177.

Powell, W. B., 1986. A local improvement heuristic for the design of less-than-truckload motor carrier networks. Transportation Science 20 (4), 246–257.

Voudouris, C., Tsang, E. P., 2003. Guided local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer, pp. 185–218.

Wieberneit, N., 2008. Service network design for freight transportation: a review. OR Spectrum 30, 77–112.